IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of: Jeffrey G. Cheng et al.	Examiner: Philip A. Guyton
Application No.: 10/672,180	Group Art Unit: 2113
Filed: September 26, 2003	Docket No.: 00100.03.0032
For: **METHOD AND APPARATUS FOR MONITORING AND RESETTING A CO-PROCESSOR**	

APPEAL BRIEF PURSUANT TO 37 C.F.R. § 41.37

Dear Sir:

Appellant submits this brief further to the Pre-Appeal Brief Request for Review filed April 30, 2008, and the Notice of Panel Decision from Pre-Appeal Brief Review dated May 30, 2008, in the above-identified application.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

ATI Technologies ULC is the real party in interest in this appeal by virtue of an executed assignment from the named inventors of their entire interest to ATI Technology Inc., a corrective assignment and an executed name change. The assignment evincing such ownership interest was recorded on September 26, 2003, in the United States Patent and Trademark Office at Reel 014543, Frame 0775. A corrective assignment was submitted for recordation on September 2, 2008 in the United States Patent and Trademark Office. The Name Change was submitted for recordation on September 2, 2008 in the United States Patent and Trademark Office.

II. RELATED APPEALS AND INTERFERENCES

To Appellant's knowledge, there are no prior or pending appeals, judicial proceedings or interferences which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this pending appeal.

III. STATUS OF CLAIMS

Claims 1-2, 4-7, 9-15, 17-19, 21-35 are pending. Claims 14-15, 17-19 and 21-23 are allowed. Claims 3, 8, 16 and 20 are cancelled. Claims 4, 9, 27 and 31 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. Claims 1-2, 5-7, 10-13, 24-26, 28-30 and 32-35 are rejected. A copy of the pending, non-cancelled claims is attached in Appendix A. Each of the rejected claims is appealed in this appeal brief. As provided in the Argument section hereof, claims 2, 6-7, 12-13 and 24 stand or fall together with independent claim 1, claims 28, 30, 32 and 34 stand or fall together with independent claim 26, claims 29 and 33 stand or fall together with claim 35, and claims 5 and 10 stand or fall together with independent claim 11.

IV. STATUS OF AMENDMENTS

Appellant did not file an amendment following the final Office action dated October 30, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

A host processor is capable of optimizing the processing of graphics data and video information by using a graphics processor and related software. (Specification at ¶ 2.) Such graphics processing is commonly referred to as graphics rendering. *Id.* The host processor communicates with the graphics processor and effectuates drawing commands and graphics rendering by executing a piece of software called a graphics driver. *Id.* Because of the complexity of hardware and software used in and to support accelerated graphics rendering, graphics processors tend to be susceptible to various rendering errors such as hangs. (*Id.* at ¶ 3.) A hang is where a system or process becomes non-responsive, such as where a processor hangs up due to the processor stalling, locking up, getting stuck in an infinite loop, etc. *Id.* Such hangs may cause data loss or may force operations to halt on an entire system. *Id.*

Conventional methods have been developed to detect errors on an operating system level (e.g., at the graphics driver level), which may be indicative of a hang in a graphics processor. (*Id.* at ¶¶ 4-6.) One such method is to periodically implement a watchdog timer that (a) measures the time spent in a graphics driver by performing a spin-loop and (b) waits for the graphics processor to respond. (*Id.* at ¶ 4.) The detection of an infinite loop is indicative that the graphics processor is likely hung. *Id.* Once a timeout expires without a response, the operating system either stops the graphics driver and prompts the user to restart the system or unloads the accelerated graphics driver and loads a standards VGA driver to continue graphics rendering operations in a non-accelerated mode. (*Id.* at ¶¶ 4, 6.)

Such conventional methods, however, are unable to differentiate between what appears to be a hang in the graphics driver and an actual hang. (*Id.* at ¶ 5.) For example, if a large amount of rendering commands are being processed by a graphics processor, the graphics processor may provide a relatively slow response to the spin-loop. *Id.* In certain instances, such a slow

response can be improperly interpreted as a hang in the graphics processor, thereby requiring a system reboot or a switch to non-accelerated graphics rendering. (*Id.* at ¶¶ 5-6.)

With reference to Figure 1 of Appellant's application, the Specification discloses processor 100 coupled to co-processor 102. *(Id.* at ¶ 33.) In one embodiment, processor 100 is a host processor and co-processor 102 is a graphics processor. (*Id.* at ¶ 41.) Processor 100 includes circuit 104 comprising a hang detector module 106 and a selective processor reset module 108. (*Id.* at ¶ 34.) The hang detector module 106 examines co-processor 102 to determine if it is hung. *Id.* Unlike the conventional methods that use a watchdog timer and that may generate false negatives, Appellant's Specification provides for the detection of a hang by detecting a discrepancy between a current state of the co-processor 102 and a current activity of the co-processor 102. (*Id.* at ¶ 49.)

In one embodiment, the hang detector module 106 determines a current state of the co-processor 102 by determining if the co-processor 102 is currently executing instructions or busy. (*Id.* at ¶ 35.) For example, the hang detector module 106 may first check to see if a busy flag (e.g., data in one or more storage elements) is set. (*Id.* at ¶¶ 36, 41.) If a busy flag is not set, the hang detector module 106 returns a no-hang notification signal to the selective processor reset module 108 because a co-processor that is not executing any instructions is unlikely to be experiencing a hang. (*Id.* at ¶¶ 36, 26.) However, if a busy flag is determined to be set, the hang detector module 106 then determines if a current activity of the co-processor 102 is consistent with the current state of the co-processor 102 (indicative of a no hang condition) or whether there is a discrepancy between the current state and the current activity of the co-processor 102 (indicative of a hang condition). (*Id.* at ¶ 35-36.)

The hang detection module 106 is capable of detecting the current activity of co-processor 102 by examining corresponding registers or other storage elements associated with the co-processor 102 to determine whether activity in those registers reflects the actual processing of instructions. *Id.* In one embodiment, corresponding registers are examined before and after a suitable wait period to ensure the co-processor 102 has an opportunity to process the instructions and alter the data in the registers. (*Id.* at ¶ 36.) In this embodiment, a comparison between the two sets of register contents (i.e., on either side of the wait period) can be performed by the hang detection module 106 to detect a difference in the data. *Id.*

If a difference (i.e., activity) is detected, the co-processor 102 is not hung because the current activity of the co-processor 102 is consistent with the current state of the co-processor, and a no-hang notification signal may be set to the selective processor reset module. *Id.* If no difference or activity is detected, the co-processor 102 is hung <u>because a discrepancy between a current state of the co-processor 102 and a current activity of the co-processor 102 has been detected</u>. (*Id.* at 49.) A discrepancy is said to exist because the state of the co-processor is "busy" (i.e., the co-processor is executing instructions), but the co-processor's activity is <u>not consistent with</u> the busy state of the co-processor 102 (i.e., the co-processor's registers do not show a difference in stored data). (*Id.* at ¶¶ 35-36, 41, 48-49.) Following the detection of the hang, the hang detection module 106 returns a hang notification signal 110 to the selective processor reset module 108. (*Id.* at ¶ 36.) When the selective processor reset module 108 receives the hang notification signal 110, indicating that a hang has occurred in co-processor 102, the selective processor reset module 108 selectively resets co-processor 102 without resetting the processor 100. *Id.*

Turning to the claims, Appellant's claims 1, 26 and 35 are representative of the features and aspects described above. For example, claim 1 requires:

> A circuit for monitoring and resetting a co-processor comprising … a hang detector module operative to detect a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and a current activity of the co-processor. (Emphasis added.)

Similarly, claim 26 requires:

> A circuit for monitoring and resetting a co-processor comprising … a hang detector module operative to detect a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and data in one or more storage elements associated with the co-processor, wherein the data in the one or more storage elements represents a current activity of the co-processor. (Emphasis added.)

And, claim 35, which is dependent upon claim 1, further adds the following requirement:

> wherein the discrepancy [between the current state and current activity of the co-processor] is detected by comparing data representing a current state of the co-processor with data representing a current activity of the co-processor. (Emphasis added.)

In one embodiment and with reference to FIG. 3, circuit 104 may include, among other things, halt communications module 300. (*Id.* at ¶ 44.) Halt communications module 300 is used to halt command communications with co-processor 102 in response to receiving hang notification signal 110 from hang detector module 106 indicating a hang in the co-processor 102. The halt communications module 300 may be used to stop the sending of instructions to the co-processor 102. *Id.* In one embodiment, halt communications module 300 may implement this task by setting a send flag, a receive flag and/or a busy flag, each associated with communication between the processor 100 and the co-processor 102, to "off". *Id.* By setting a send flag to "off", processor 100 is prevented form sending instructions to the co-processor 102. *Id.* By

setting a receive flag or busy flag to "off", processor 100 does not expect to receive any return signals, data or the like from co-processor 102. *Id.*

Claim 11 is illustrative of this feature. Claim 11 requires a circuit for monitoring and resetting a co-processor comprising, among other things:

> a hang detector module operative to detect a hang in a co-processor; [and]
> a halt communications module operative to halt executable instruction communications with the co-processor, in response to detecting a hang in the co-processor. (Emphasis added.)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-2, 5-7, 10-11, 24-26, 28-30 and 32-35 stand rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,742,139 to Forsman et al. ("Forsman"). Claims 12 and 13 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Forsman in view of U.S. Patent Application No. 2002/0093505 to Hill et al. ("Hill").

VII. ARGUMENT: FORSMAN FAILS TEACH EACH AND EVERY LIMITATION OF CLAIMS 1-2, 5-7, 10-11, 24-26, 28-30 AND 32-35

To establish a *prima facie* case of anticipation, each and every limitation of a claim must be found, either expressly or inherently, in a single prior art reference. MPEP § 2131. Because the Forsman fails to teach each and every limitation of claims 1-2, 5-7, 10-13, 24-26, 28-30 and 32-35[1], the rejections to these claims should be reversed and the claims should be allowed.

A. CLAIMS 2, 6-7, 12-13 AND 24 STAND OR FALL TOGETHER WITH INDEPENDENT CLAIM 1

Claims 1-2, 6-7 and 24 stand rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by Forsman. Claims 12 and 13 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Forsman in view of Hill. Claims 2, 6-7, 12-13 and 24 stand or fall together with claim 1. Claim 1 requires a circuit for monitoring and resetting a co-processor comprising:

> a hang detector module operative to detect a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and a current activity of the co-processor; and
>
> a selective processor reset module operative to selectively reset the co-processor without resetting a processor, in response to detecting a hang in the co-processor. (Emphasis added).

The final Office action mailed October 30, 2007 (the "Final Office Action") alleges that Forsman discloses all aspects of Appellant's claim 1. (Final Office Action, p. 2.) Appellant disagrees because Forsman is limited to a system and method that (1) resets a service processor when a host processor fails to detect a heartbeat signal from the service processor; or (2) resets a service processor when (a) a host processor fails to detect a heartbeat signal from the service

[1] Claims 12 and 13 are rejected under 35 U.S.C. § 103(a). However, because Forsman does not teach what is alleged, the obviousness rejections are also improper and should be reversed.

processor and (b) the host processor determines that conditions do not exist that preempt the host processor from resetting the service processor. In the first instance, Forsman fails to detect any sort of discrepancy between two things such as a current state and a current activity of a co-processor. In the second instance, Forsman resets the service processor only after determining a consistent condition between the lack of heartbeat signals and a lack of preempting conditions.

(i) *The Forsman Reference*

Forsman appears to be directed to a system and method for resetting a service processor and reestablishing communications between a host processor and a service processor after the service processor has ceased to function correctly. (Forsman, Title & Abstract.) Forsman teaches that "[i]n a proper running state … [h]ost 202 ("host processor") and service processor 204 … exchange heartbeat signals 206." (*Id.*, Col. 4, ll. 9-12.) In a first embodiment, if the host processor 202 fails to detect a heartbeat signal 206 from the service processor 204, the service processor 204 is not functioning correctly and must be reset. (*Id.*, Col. 4, ll. 13-15, 25-35.) Specifically, a hard reset of the service processor 203 is initiated in a way that does not disturb host processor 202 usage of shared resources. *Id.*

In a second embodiment where the host processor 202 initiates the communication recovery or hard reset of the service processor 204, host processor 202, after detecting alack of heartbeat signals, "checks the status portion of the status/control register 208 in hardware logic 212 to determine if conditions exist that preempt host [processor] 202 from resetting service processor 204." (*Id.*, Col. 4, ll. 36-40.) Examples of conditions that may preempt the resetting of the service processor 204 include when the service processor 204 is: in a special debug mode, in the process of handling a critical event, and attempting to recover from a self detected error. (*Id.*, Col. 4, ll. 40-45.) If no status exceptions (i.e. conditions that preempt the host processor

202 from resetting the service processor 204) exist, the host processor then provides a warning to the service processor 204 that a hard reset is about to commence. (*Id.*, Col. 4, ll. 46-50.) If the service processor 204 acknowledges this warning and agrees to being reset or if the service processor 204 does not respond within a timeout period, the host processor 202 resets the service processor 204. (*Id.*, Col. 4, ll. 46-65.)

In the former case, Forsman relies only on the detection, by the host processor 202, of a lack of heartbeat signals from the service processor 204 before resetting the service processor 204. In other words, Forsman fails to teach, expressly or inherently, a detection of any type of discrepancy between a current state and a current activity of a co-processor. At best, Forsman appears to teach a detection of a current activity (i.e., the lack of heartbeat signals from a service processor 204) and associates this current activity with a "not functioning correctly" condition. In the latter case, Forsman relies not on a detection of a discrepancy, but on the detection of a consistency between the lack of heartbeat signals and the lack of any preemptive conditions or status exceptions in determining that the service processor is not functioning correctly and should be reset. Because Forsman fails to teach the detection of a discrepancy between a current state of a co-processor and a current activity of the co-processor, as required by claim 1, the rejection is in error and must be withdrawn.

(ii) *The Final Office Action's Rejection*

Ignoring the plain teachings in Forman, the Final Office Action provides the following explanation of its rejection of claim 1.

> The cited portions of Forsman disclose a system in which a host processor and service processor exchange heartbeat signals (column 4, lines 9-12). The heartbeat signals indicate that a service processor is active and working correctly (column 1, lines 35-37). In other words, the heartbeat signal is an indication of the current activity of the processor. In order for an abnormality to be found, the status of the service processor must be normal (column

> 4, lines 36-45). Thus, a discrepancy exists, and the host [processor] resets the service processor and returns it to its original state and activity where it can continue to process heartbeat signals (column 4, lines 46-50). (Final Office Action, p. 9.)
>
> Furthermore, in Forsman, the absence of heartbeat signals does not automatically indicate a service processor failure, as the status register may indicate an exceptional condition …. Thus, applicant's interpretation that at the time of failure to detect heartbeat signals the current state of the processor is not active and working correctly is incorrect. Instead the current status is actually determined by status of the service processor at the moment that failure of heartbeat signal detection occurs. (*Id.*, p. 10.)

Appellant submits that the above comments constitute clear error on behalf of the Examiner because it manipulates the teachings of Forsman in an attempt to provide a *prima facie* case of anticipation against Appellant's claim 1, where none exists.

Specifically, the Examiner states that the exchange or lack of heartbeat signals is akin to Appellant's claimed current activity of a co-processor. (Final Office Action, p. 9.) Citing to column 4, lines 36-45 of Forsman, the Examiner then states that "[i]n order for an abnormality to be found, the current state of the service processor must be normal." *Id.* However, this portion of Forsman does not stand for the proposition set forth by the Examiner. Instead, this portion of the reference, as properly described above, discusses an embodiment of Forsman where the host processor 202 checks to see if there are any conditions that may preempt the host processor 202 from resetting the service processor 204.

Appellants are unable to find any teaching, express or implicit, in Forsman that stands for the proposition put forth by the Examiner. In fact, Appellants submit that Forsman appears to teach the opposite: when the host processor 202 fails to detect heartbeat signals, the current state of the service processor 204 is working improperly. (Forsman, Col. 4, ll. 13-15.) This current state is consistent with the current activity of the service processor 204 (i.e., not sending heartbeat signals).

At best, it appears that the Examiner is manipulating Forsman to suggest that a discrepancy of any kind is detected. As described above, Forsman is limited to: (1) resetting a service processor when a host processor fails to detect a heartbeat signal from the service processor; or (2) resetting a service processor when (a) a host processor fails to detect a heartbeat signal from the service processor and (b) the host processor determines that conditions do not exist that preempt the host processor from resetting the service processor. In the first instance, Forsman fails to detect any sort of discrepancy between two things such as a current state and a current activity of a co-processor. Instead, Forsman immediately resets the service processor. (Forsman, Col. 4, ll. 25-35.) In the second instance, Forsman resets the service processor only after determining a consistent condition between the lack of heartbeat signals and a lack of preempting conditions. (Forsman, Col. 4, ll. 25-65.) Each of the reasons presented immediately above demonstrate the error in the Examiner's rejection of claim 1.

B. CLAIMS 28, 30, 32 AND 34 STAND OR FALL TOGETHER WITH INDEPENDENT CLAIM 26

Claims 26, 28, 30, 32 and 34 stand rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by Forsman. Claims 28, 30, 32 and 34 stand or fall together with independent claim 26. Claim 26 requires a circuit for monitoring and resetting a co-processor comprising:

> a hang detector module operative to detect a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and data in the one or more storage elements associated with the co-processor, wherein the data in the one or more storage elements represents a current activity of the co-processor; and
>
> a selective processor reset module operative to selectively reset the co-processor without resetting a processor, in response to detecting a hang in the co-processor. (Emphasis added).

The Final Office Action alleges that Forsman discloses all aspects of Appellant's claim

26. (Final Office Action, pp. 2, 5.) Appellant disagrees for the same reasons set forth above. Namely, Forsman is limited to a system and method that (1) resets a service processor when a host processor fails to detect a heartbeat signal from the service processor; or (2) resets a service processor when (a) a host processor fails to detect a heartbeat signal from the service processor and (b) the host processor determines that conditions do not exist that preempt the host processor from resetting the service processor. In the first instance, Forsman fails to detect any sort of discrepancy between two things such as a current state and a current activity of a co-processor. In the second instance, Forsman resets the service processor only after determining a consistent condition between the lack of heartbeat signals and a lack of preempting conditions.

(i) *The Final Office Action's Rejection*

Once again ignoring the plain teachings in Forman, the Final Office Action provides the following explanation of its rejection of claim 26.

> Regarding claims 26, 30 and 36 … Forsman teaches that wherein the status/control register is checked after a failure to detect heartbeat signals (column 4, lines 36-45). If no status exceptions are present, then the processor is reset (column 4, lines 46-50). Examples of status exceptions include when the service processor is in a special debug mode or when the service processor is in the process of handling a critical event (column 4, lines 40-44). Both of these are indications that the service processor is not in a hang condition. Thus, checking of the status register is part of the detection of a hang in the service processor. (Final Office Action, p. 10.)

Initially, Appellant submits that the checking of the status register to determine if any conditions (e.g., status exceptions) exist that preempt the host processor 202 from resetting the service processor 204 appears to be part of detecting a hang in the service processor 202 in the Forsman embodiment where the host processor initiates the communications recovery or resetting of the service processor 204. (Forsman, Col. 4, ll. 36-50.) In the more general embodiment, described

above as the first embodiment, the checking of the status register is not performed. (*Id.*, Col. 4, ll. 25-35.)

Appellant further submits that the Examiner's own rejection supports Appellant's argument. The Examiner clearly states that examples of status exceptions are indications that the service processor is not in a hang condition. (Final Office Action, p. 10.) The corollary of this statement is also true: The lack of status exceptions are indications that the service processor 204 is in a hang condition. (Forman, Col. 4, ll. 36-50.) Thus, as noted above, in the embodiment where the host processor 202 resets the service processor 204, Forsman looks for consistency between the lack of heartbeat signals and the lack of status exceptions. The lack of heartbeat signals tends to indicate a hang condition, while the lack of status exceptions confirms the hang condition. Forsman does not appear to teach the detection of a discrepancy between a current state of the co-processor and data in one or more storage elements associated with the co-processor, wherein the data in the one or more storage elements represents a current activity of the co-processor. This reason demands the withdrawal of rejection against claim 26.

C. CLAIMS 29 AND 33 STAND OR FALL TOGETHER WITH CLAIM 35

Claims 29, 33 and 35 stand rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by Forsman. Claims 29 and 33 stand or fall together with claim 35. Claim 35 reads: the circuit of claim 1, wherein:

> The discrepancy is detected by comparing data representing a current state of the co-processor with data representing a current activity of the co-processor. (Emphasis added).

The Final Office Action alleges that Forsman discloses all aspects of Appellant's claim 1. (Final Office Action, p. 2, 7.) Appellant disagrees for the same reasons set forth above in subsections 7(A) and 7(B). Appellant further notes that Forsman appears to be silent as to comparing data representing a current state of the co-processor with data representing a current activity of the co-

processor. Instead, Forsman appears to teach recognizing the lack of a heartbeat signal as a hang or recognizing the lack of a heartbeat signal and ensuring that a hang exists by examining the contents of a status/control register for the purpose of detecting a consistency (not a discrepancy) indicative of a hang condition.

(ii) *The Final Office Action's Rejection*

In its rejection of claim 35, the Final Office Action merely points to column 4, lines 25-50. However, these are the same lines fully explained above with respect to the claims 1 and 26. As noted, Forsman does not appear to be capable of comparing data representing a current state of a co-processor with data representing a current activity of a co-processor. Instead, Forsman is limited to detecting the lack of a heartbeat signal (i.e., the lack of data) and either: (1) resetting the service processor based on the lack of a heartbeat signal or (2) resetting the service processor after confirming that the service processor is hung by not finding any status exceptions in a status/control register.

D. CLAIMS 5 AND 10 STAND OR FALL TOGETHER WITH INDEPENDENT CLAIM 11

Claims 5 and 10-11 stand rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by Forsman. Claims 5 and 10 stand or fall together with independent claim 11. Claim 11 requires a circuit for monitoring and resetting a co-processor comprising, among other things:

> a hang detector module operative to detect a hang in the co-processor; [and]
>
> a halt communications module operative to halt executable instruction communications with the co-processor, in response to detecting a hang in the co-processor. (Emphasis added).

The Final Office Action alleges that Forsman discloses all aspects of Appellant's claim 11. (Final Office Action, pp. 2, 4.) Appellant disagrees because Forsman merely provides for a timeout period after determining that a hard reset of the service processor 204 is required, whereby during the timeout period, the host processor 202 waits to receive an acknowledgement that service processor 204 is ready to be reset. (Forsman, Col. 5, ll.10-23.) If no such acknowledgement is received, the host processor 202 resets the service processor 204 after the timeout period expires. Forsman appears silent as to a halt communication module operative to halt execution instruction communications with a co-processor.

(i) *The Forsman Reference*

Forsman clearly provides that after the host processor 202 determines that there are no status exceptions that preempt the host processor 202 from resetting the service processor 204, "then the host [processor] sends a signal to the service processor warning the service processor that a hard reset is about to occur." (Forsman, Col. 5, ll. 10-14.) "The host [processor] then determines whether an acknowledgement has been received or a timeout has occurred." (*Id.*, Col. 5, ll. 16-17.) "The acknowledgement indicates that the service processor has received the warning and is ready to be reset. The timeout period is a predefined interval of time that the host must wait for a response from the service processor before assuming that the service processor is not going to respond." (*Id.*, Col. 5, ll. 17-22.) "Once the acknowledgement has been received or the timeout period occurred, the host causes a hard reset of the service processor." (*Id.*, Col. 5, ll. 23-26.)

As explained, Forsman merely provides a timeout period during which the service processor 204 has an opportunity to signal to the host processor 202 that is ready to be reset.

Forsman is silent as to a halt communication module operative to halt executable instruction communications with the co-processor.

(ii) *The Final Office Action's Rejection*

Once again ignoring the plain teachings in Forman, the Final Office Action provides the following explanation of its rejection of claim 11.

> With regard to claim 11, … Applicant asserts that the disclosure of waiting a predetermined timeout period for the service processor to respond to a reset warning does not teach halting of command communications with the co-processor because communication is possible. However, the examiner maintains that this is an incorrect interpretation of the claim. In Forsman, although communication is possible, during the period that the host is waiting for acknowledgement, communication has been halted (column 5, lines 10-22 and figure 2, steps 304-308). In this period of time that the host is waiting, whether an acknowledgement is received or the process times-out, there is not other communication between the host and the service processor. Thus, all communications, including executable instruction communications are halted. (Final Office Action, p. 11.)

Appellant submits that the Examiner comments above constitute clear error because it ignores claim language. Claim 11 requires, among other things, a circuit comprising a halt communications module that is operative to halt executable instruction communications. Forsman neither teaches nor describes any such module. The Examiner improperly assumes that something like a halt communications module must exist in Forsman because communication has allegedly been halted during the timeout period. This is improper because: (1) Forsman does not state that communication has been halted; and (2) Forsman does not provide, teach or even suggest that any halt communication module exists to halt executable instruction communications with the co-processor. In fact, Forsman expressly states that communication from the service processor 202 to the host processor 204 may exist by way of an

acknowledgement message. Because Forsman does not teach what is suggested, claim 11 is in proper condition for allowance.

VIII. CONCLUSION

For the reasons advanced above, Appellant submits that the Examiner erred in rejecting pending claims 1-2, 5-7, 10-13, 24-26, 28-30 and 32-35 and respectfully requests reversal of the decision of the Examiner.

Respectfully submitted,

Date: September 2, 2008

By: /Christopher J. Reckamp/
Christopher J. Reckamp
Registration No. 34,414

Vedder Price P.C.
222 N. LaSalle Street
Chicago, Illinois 60601
PHONE: (312) 609-7599
FAX: (312) 609-5005

CLAIMS APPENDIX

CLAIMS ON APPEAL

1. (Previously Presented) A circuit for monitoring and resetting a co-processor comprising:

a hang detector module operative to detect a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and a current activity of the co-processor; and

a selective processor reset module operative to selectively reset the co-processor without resetting a processor, in response to detecting a hang in the co-processor.

2. (Previously Presented) The circuit of claim 1 wherein an operating system executes on the processor.

3. (Cancelled)

4. (Previously Presented) The circuit of claim 1 wherein the discrepancy is detected by detecting the current state to be busy, by detecting a busy flag to be set, and detecting no progress on current activity, by detecting the same contents in a co-processor register as examined before and after a wait period.

5. (Previously Presented) The circuit of claim 1 further comprising:

a halt communications module operative to halt command communications with the co-processor, in response to detecting a hang in the co-processor;

a reset check module operative to detect if the co-processor has been successfully reset, in response to the resetting of the co-processor; and

a restart communications module operative to restart command communications with the co-processor, in response to detecting that the co-processor has been successfully reset.

6. (Previously Presented) A method of monitoring and resetting a co-processor comprising:

detecting a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and a current activity of the co-processor; and

selectively resetting the co-processor without resetting a processor, in response to detecting a hang in the co-processor.

7. (Original) The method of claim 6 wherein an operating system executes on the processor.

8. (Cancelled)

9. (Previously Presented) The method of claim 6 wherein the discrepancy is detected by detecting the current state to be busy, by detecting a busy flag to be set, and detecting no progress on current activity, by detecting the same contents in a co-processor register as examined before and after a wait period.

10. (Previously Presented) The method of claim 6 further comprising:

halting command communications with the co-processor, in response to detecting a hang in the co-processor;

detecting if the co-processor has been successfully reset, in response to the resetting of the co-processor; and

restarting command communications with the co-processor, in response to detecting that the co-processor has been successfully reset.

APPENDIX A

11. (Previously Presented) A circuit for monitoring and resetting a co-processor comprising:

a hang detector module operative to detect a hang in the co-processor;

a halt communications module operative to halt executable instruction communications with the co-processor, in response to detecting a hang in the co-processor;

a selective processor reset module operative to selectively reset the co-processor without resetting a processor, in response to detecting a hang in the co-processor;

a reset check module operative to detect if the co-processor has been successfully reset, in response to the resetting of the co-processor; and

a restart communications module operative to restart executable instruction communications with the co-processor, in response to detecting that the co-processor has been successfully reset.

12. (Original) The circuit of claim 11 wherein the processor is a host processor and the co-processor is a graphics processor.

13. (Original) The circuit of claim 12 wherein the hang detector module detects the hang in the graphics processor by detecting a discrepancy between a current state of the graphics processor and a current activity of the graphics processor.

14. (Allowed) A system for monitoring and resetting a co-processor comprising:

a processor;

a co-processor; and

a memory containing instructions including:

hang detector module instructions operative to cause the processor to detect a hang in the co-processor by detecting the current state to be busy, as reflected in a busy flag, and detecting no progress on current activity, as reflected in the absence of co-processor register activity; and

selective processor reset module instructions operative to cause the processor to selectively reset the co-processor without resetting the processor, in response to detecting a hang in the co-processor.

15. (Allowed) The system of claim 14 wherein the processor is a host processor and the memory contains operating system instructions.

16. (Cancelled)

17. (Allowed) The system of claim 14 further comprising:

reset check module instructions operative to cause the processor to detect if the co-processor has been successfully reset, in response to the resetting of the co-processor.

18. (Allowed) The system of claim 14 further comprising:

halt communications module instructions operative to cause the processor to halt command communications with the co-processor, in response to detecting a hang in the co-processor; and

restart communications module instructions operative to cause the processor to restart command communications with the co-processor, in response to detecting that the co-processor has been successfully reset.

19. (Allowed) A system for monitoring and resetting a co-processor comprising:

a host processor;

a graphics processor; and

a memory containing instructions including:

hang detector module instructions operative to cause the processor to detect a hang in the graphics processor by detecting the current state of the graphics processor to be busy, as reflected in a graphics processor busy flag, and detecting the current activity of the graphics processor to be idle, as reflected in the absence of graphics processor register activity; and

selective processor reset module instructions operative to cause the processor to selectively reset the graphics processor without resetting the host processor, in response to detecting a hang in the graphics processor.

20. (Cancelled)

21. (Allowed) The system of claim 19 further comprising:

halt communications module instructions operative to cause the processor to halt rendering command communications with the graphics processor, in response to detecting a hang in the graphics processor;

reset check module instructions operative to cause the processor to detect if the graphics processor has been successfully reset, in response to the resetting of the graphics processor; and

restart communications module instructions operative to cause the processor to restart rendering command communications with the graphics processor, in response to detecting that the graphics processor has been successfully reset.

22. (Allowed) A system for monitoring and resetting a co-processor comprising:

a host processor;

a graphics processor; and

a memory containing instructions including:

hang detector module instructions operative to cause the processor to detect a hang in the graphics processor by detecting the current state to be busy, by detecting a busy flag to be set, and detecting no progress on current activity, by detecting the same contents in a co-processor register as examined before and after a wait period;

halt communications module instructions operative to cause the processor to halt rendering command communications with the graphics processor by setting a send flag to an off state and setting a receive flag to an off state, in response to detecting a hang in the graphics processor;

save snapshot module instructions operative to cause the processor to save a snapshot of the hardware and software status including any one or more of the following: graphics register data, graphics command queue data, chipset info, and AGP bus status, in response to the detecting a hang in the graphics processor;

selective processor reset module instructions operative to cause the processor to selectively reset the graphics processor without resetting the host processor, in response to detecting a hang in the graphics processor;

reset check module instructions operative to cause the processor to detect if the graphics processor has been successfully reset, in response to the resetting of the graphics processor;

display mode switch module instructions operative to cause the processor to perform a display mode switch, in response to the selectively resetting of the graphics processor;

functioning check module instructions operative to cause the processor to detect if the graphics processor is fully functioning, in response to the resetting of the graphics processor;

restart communications module instructions operative to cause the processor to restart rendering command communications with the graphics processor by setting the send flag to an on state, in response to detecting that the graphics processor has been successfully reset; and

software rendering module instructions operative to cause the processor to dynamically switch to software rendering mode, in response to detecting any one or more of the following: a detection that the graphics processor has not been successfully reset and a detection that the graphics processor is not fully functioning.

23. (Allowed) The system of claim 22 further comprising:

hang resolved prompt module instructions operative to cause the processor to display a prompt indicating that a hang was detected and resolved, in response to detecting that the graphics processor is fully functioning;

report send prompt module instructions operative to cause the processor to display a prompt requesting an input as to whether a user wants to have an error report sent to a remote location, in response to detecting a hang in the graphics processor;

report send module instructions operative to cause the processor to send an error report to a remote location including the hardware and software status, in response to receiving a user request to send an error report to a remote location; and

hang unresolved prompt module instructions operative to cause the processor to display a prompt indicating that a hang was detected and cannot be resolved without a reset of the host processor being performed, in response to detecting one or more of the following: the graphics processor was not successfully reset and the graphics processor is not fully functioning.

24. (Previously Presented) A memory containing instructions executable on a processor that causes the processor to:

detect a hang in a co-processor by detecting a discrepancy between a current state of the co-processor and a current activity of the co-processor; and

selectively reset the co-processor without resetting the processor, in response to detecting a hang in the co-processor.

25. (Original) The memory of claim 24 further including instructions that causes the processor to:

detect if the co-processor has been successfully reset, in response to the resetting of the co-processor.

26. (Previously Presented) A circuit for monitoring and resetting a co-processor comprising:

a hang detector module operative to detect a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and data in one or more storage elements associated with the co-processor, wherein the data in the one or more storage elements represents a current activity of the co-processor; and

a selective processor reset module operative to selectively reset the co-processor without resetting a processor, in response to detecting a hang in the co-processor.

27. (Previously Presented) The circuit of claim 26 wherein:

the one or more storage elements includes a co-processor register; and

the discrepancy is detected by detecting the current state to be busy, by detecting a busy flag to be set, and detecting no progress on current activity, by detecting the same contents in a co-processor register as examined before and after a wait period.

28. (Previously Presented) The circuit of claim 26 wherein the hang detector module is operative to determine if the data in the one or more storage elements reflects processing of instructions by the co-processor.

29. (Previously Presented) The circuit of claim 26, wherein the current state of the co-processor is represented by data stored in the one or more storage elements associated with the co-processor.

30. (Previously Presented) A method of monitoring and resetting a co-processor comprising:

detecting a hang in the co-processor by detecting a discrepancy between a current state of the co-processor and data in one or more storage elements associated with the co-processor, wherein the data in the one or more storage elements represents a current activity of the co-processor; and

selectively resetting the co-processor without resetting a processor, in response to detecting a hang in the co-processor.

31. (Previously Presented) The method of claim 30 wherein:

the one or more storage elements includes a co-processor register; and

the discrepancy is detected by detecting the current state to be busy, by detecting a busy flag to be set, and detecting no progress on current activity, by detecting the same contents in a co-processor register as examined before and after a wait period.

32. (Previously Presented) The method of claim 30 further comprising determining if the data in the one or more storage elements reflects processing of instructions by the co-processor.

33. (Previously Presented) The method of claim 30, wherein the current state of the co-processor is represented by data stored in the one or more storage elements associated with the co-processor.

34. (Previously Presented) A memory containing instructions executable on a processor that causes the processor to:

detect a hang in a co-processor by detecting a discrepancy between a current state of the co-processor and data in one or more storage elements associated with the co-processor, wherein the data in the one or more storage elements represents a current activity of the co-processor; and

selectively reset the co-processor without resetting the processor, in response to detecting a hang in the co-processor.

35. (Previously Presented) The circuit of claim 1 wherein the discrepancy is detected by comparing data representing a current state of the co-processor with data representing a current activity of the co-processor.

EVIDENCE APPENDIX

[NONE]

RELATED PROCEEDINGS

[NONE]